BCBSRI-logo

HCLTech | HCL America

HCLTech Confidential

Claims Details API

Design Document

Version # : V2.3

Released on : 10/01/2025

Author : Saikiran Reddy P

## Table of Contents

1. **General Information**

1.1 Executive Summary

1.2 Scope

1.3 Document Usage

1.4 Diagrams (Placeholders)

2. **Technical Design**

2.1 Azure Function One — Copyer (Queue Trigger)

2.1.1 Input Message Structure

2.1.2 Processing Steps

2.1.3 Outputs & Audit

2.1.4 Error Handling (Function One)

2.2 Azure Function Two — Processor (Queue Trigger)

2.2.1 Input Message Structure

2.2.2 Processing Steps

2.2.3 Outputs & Audit

2.2.4 Error Handling (Function Two)

2.3 End-to-End Sequence

3. **Error Handling and Recovery**

3.1 Error Taxonomy and Codes

3.2 Retry, DLQ, and Quarantine

3.3 ERROR\_LOG and PROCESS\_LOG usage

4. **Business Rules**

5. **Developer Implementation Plan** (Phase A–E)

5.1 Developer Checklist & Local Setup

6. **Monitoring & Operations**

7**. References**

8. **Amendment History**

Appendix A: Snowflake DDL (PENDED\_CLAIMS\_OUTBOUND + supporting tables)

Appendix B: Sample JSON Payloads (Queue, Claim, API)

Appendix C: Error Code Reference Table

Software Design Document (SDD) – PCSE: Outbound Claims Processing

# 1. General Information

## 1.1 *Executive Summary*

This document provides copying and processing claim files from Azure Blob Storage into Snowflake. The flow consists of two Azure Functions: Function One copies incoming claim files from a source container to a staging container; Function Two (Processor) reads the staged files, filters Prime-pended claims, transforms the payloads, and ingests them into Snowflake via the Member API. The design focuses on reliability, idempotency, auditability, and operational visibility.

## 1.2 *Scope*

In-Scope:

- Function One (Queue-triggered): download & copy blob to staging container; create audit entry; enqueue staging message.

- Function Two (Queue-triggered): parse staged blob; filter & transform claims; call Member API to load into Snowflake; record errors and audit.

Out-of-Scope:

- Kafka ingestion and the upstream Facets system (already existing).

- Downstream Matillion transformation and SFTP transfer (handled by other flows).

### 1.2.2 *Out-of-Scope*

• UI components  
• Manual claim processing  
• Downstream systems outside Snowflake.

- Inbound flow from Prime back to Facets.  
- Initial publishing of claims from Facets to Kafka and Blob Storage.  
- Downstream transformation by Matillion or other ETL tools beyond Snowflake staging.

## 1.3 *Document Usage*

This document serves as a reference for all stakeholders including solution architects, developers, QA engineers, and project managers. It provides a single source of truth for PCSE Outbound’s technical design. Any changes to this design must follow SDLC governance and be approved by architecture review boards.

|  |  |
| --- | --- |
| User Type | Description of Usage |
| Business Analyst | Section 1 |
| Architect | Section 2 |
| Developer | Section 2, 3 & 4 |
| Tester | Section 5 |

## 1.4 Diagrams

### 1.4.1 *Data Flow Diagram*

A diagram of a cloud and lightning

AI-generated content may be incorrect.

### 1.4.2 *Use Case Diagram*

[Placeholder for Use Case Diagram: Actor = Queue/Event, System = Function, API, DB]

For this document, detailed text explanations are provided instead of diagrams. The architecture can be summarized as: Event Grid triggers → Storage Queue → Azure Function → Blob Storage → Filtering & Transformation → Snowflake staging table → Downstream systems.

## 1.5 *Specification Sheet*

• Azure Function: Queue Trigger  
• Blob Storage: JSON source  
• API: Member API endpoint  
• Database: Snowflake with Claim Table & Error Table.

Specification details of input/output JSON, queue message structure, Snowflake schema, and logging are covered in subsequent sections.

# 2. Technical Design

### 2.1 *Azure Function One — Queue Trigger*

Function One is triggered by a Storage Queue message published by Event Grid when a new blob is created in the 'claims-raw' container. It is responsible for validating the message, downloading the blob, copying it to the 'claims-staging' container, creating an audit record, and pushing a message to the staging queue for Function Two.

#### 2.1.1 Input Message Structure (Event Grid shape)

{  
 "topic":

"/subscriptions/{sub}/resourceGroups/{rg}/providers/Microsoft.Storage/storageAccounts/{sa}",  
 "subject": "/blobServices/default/containers/claims-raw/blobs/2025/09/25/claimsfile.json",  
 "eventType": "Microsoft.Storage.BlobCreated",  
 "id": "event-id-123",  
 "data": {  
 "api": "PutBlob",  
 "clientRequestId": "abcd-1234",  
 "requestId": "req-5678",  
 "eTag": "0x8DDFC7792",  
 "contentType": "application/json",  
 "contentLength": 38774,  
 "url": "https://{sa}.blob.core.windows.net/claims-raw/2025/09/25/claimsfile.json"  
 }  
}

#### 2.1.2 Processing Steps

1. Deserialize the queue message and validate 'data.url'.

2. Validate blob extension (.json/.ndjson/.csv) and size thresholds (configurable).

3. Download blob content using Azure Blob SDK (BlobClient).

4. Optionally validate top-level JSON structure (array/object).

5. Copy blob stream to destination container 'claims-staging' using a deterministic path (e.g., /{yyyy}/{mm}/{dd}/{batchId}-{filename}).

6. Insert AUDIT\_CLAIM\_FILE record with TransactionStatus = 'N' and store metadata (BATCH\_ID, PROCESS\_ID).

7. Enqueue message to 'claims-staging-queue' with destination blob metadata for Function Two.

#### 2.1.3 Outputs & Audit

- Writes AUDIT\_CLAIM\_FILE row (initial TransactionStatus = 'N').

- Sends message to 'claims-staging-queue'.

- Logs operation and metrics to Application Insights and PROCESS\_LOG.

#### 2.1.4 Error Handling (Function One)

Validation errors: If the queue message is malformed or missing URL, log ERR-QMSG-001, insert ERROR\_LOG entry, and move to quarantine if appropriate.  
Blob not found: ERR-BLOB-404 — write audit entry, optionally requeue (transient) or create ticket if persistent.  
Copy failures: ERR-COPY-3001 — retry with exponential backoff; if exhausted, write to DLQ and ERROR\_LOG.  
All exceptions: capture stack trace, correlation IDs (BATCH\_ID/PROCESS\_ID), and write PROCESS\_LOG and ERROR\_LOG entries.

### 2.2 *Azure Function Two — Processor (Queue Trigger)*

Function Two processes staging blob messages. It downloads the staged file, iterates claims, filters 'Prime-pended' claims, transforms messages for Snowflake ingestion, calls the Member API endpoint and records outcomes.

#### 2.2.1 Input Message Structure (Staging Queue)

{  
 "topic": "...",  
 "subject": "/blobServices/default/containers/claims-staging/blobs/2025/09/25/batch-123-claimsfile.json",  
 "data": {  
 "url": "https://{sa}.blob.core.windows.net/claims-staging/2025/09/25/batch-123-claimsfile.json",  
 "contentType": "application/json"  
 }  
}

#### 2.2.2 Processing Steps

1. Deserialize the staging queue message and extract blob URL.

2. Download blob and parse JSON (array of claim objects).

3. For each claim:

a. Validate mandatory fields (claim\_id, member\_id).

b. Check business filter: status == 'PENDING' and pended\_reason contains 'Prime' (case-insensitive) or primePended flag == true.

c. If valid: generate CLAIM\_GUID (UUID), add ingestion\_timestamp, normalize diagnosis & procedure codes arrays; build claim object for Member API.

d. If invalid: insert record into ERROR\_LOG with appropriate error code and continue.

4. Group claims into batch payload and call Member API (HTTP POST) with authentication via Managed Identity or API Key from Key Vault.

5. On API success (200 OK): update AUDIT\_CLAIM\_FILE TransactionStatus='Y' and insert records into PENDED\_CLAIMS\_OUTBOUND with LOAD\_STATUS='SUCCESS'.

6. On API 4xx: treat as permanent failure for those payloads — insert into ERROR\_LOG; do not retry automatically unless requeued by operator.

7. On API 5xx / network errors: retry with exponential backoff; if exhausted, mark batch as failed, write ERROR\_LOG, push to DLQ, and create ServiceNow ticket if configured.

#### 2.2.3 Outputs & Audit

- Inserts rows into PENDED\_CLAIMS\_OUTBOUND for successful claims.

- Updates AUDIT\_CLAIM\_FILE with TransactionStatus and timestamps.

- Writes claim-level errors into ERROR\_LOG for manual remediation.

#### 2.2.4 Error Handling (Function Two)

Validation errors: ERR-CLAIM-VAL-001 — insert claim-level ERROR\_LOG, continue processing others.  
Member API 5xx: ERR-API-4003 — retry with backoff; if still failing, mark batch-level failure, write to ERROR\_LOG, and DLQ.  
Snowflake insert issues: ERR-SNOW-500 — log error detail, mark LOAD\_STATUS='FAILED' for affected rows.

### 2.3 End-to-End Sequence

1. Blob uploaded to claims-raw -> Event Grid publishes message to 'claims-notifications-queue'.  
2. Function One picks up message -> copies blob to claims-staging -> writes AUDIT\_CLAIM\_FILE (N) -> enqueues staging message.  
3. Function Two picks up staging message -> processes claims -> calls Member API -> writes PENDED\_CLAIMS\_OUTBOUND -> updates audit (Y/N).  
4. Operator investigates ERROR\_LOG entries and re-enqueues failed claims if applicable.

# 3. Error Handling and Recovery

### 3.1 *Error Taxonomy and Codes*

Error codes below include both a string and a numeric mapping for clarity and operational automation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numeric Code | String Code | Category | Description | Suggested Action |
| 2001 | ERR-QMSG-001 | Queue Message | Invalid or malformed queue message | See remediation steps in 3.3 |
| 2002 | ERR-BLOB-404 | Blob | Source blob not found | See remediation steps in 3.3 |
| 2003 | ERR-BLOB-403 | Blob | Blob access denied (permissions) | See remediation steps in 3.3 |
| 3001 | ERR-COPY-3001 | Copy | Failed to copy blob to staging | See remediation steps in 3.3 |
| 1001 | ERR-CLAIM-VAL-001 | Validation | Claim missing mandatory fields or malformed | See remediation steps in 3.3 |
| 4003 | ERR-API-4003 | API | Member API returned 5xx/server error | See remediation steps in 3.3 |
| 5001 | ERR-SNOW-500 | Snowflake | Insert/load failure in Snowflake | See remediation steps in 3.3 |
| 6001 | ERR-DLQ-6001 | DLQ | Message moved to DLQ after exceeding retries | See remediation steps in 3.3 |
| 7001 | ERR-AUDIT-100 | Audit | Failed to write audit record | See remediation steps in 3.3 |

### 3.2 *Retry, DLQ, and Quarantine*

Retry policies: Transient errors are retried with exponential backoff (e.g., 2s, 4s, 8s). Configure host.json maxDequeueCount=5. After max retries, messages go to DLQ where a DLQ handler processes them.

Quarantine: Files failing schema-wide validation are moved to 'claims-quarantine' container and flagged in AUDIT\_CLAIM\_FILE for manual review.

### 3.3 *ERROR\_LOG and PROCESS\_LOG usage*

PROCESS\_LOG: function-level operational logs (INFO/WARN/ERROR). ERROR\_LOG: claim-level or batch-level failure records with ERROR\_CODE, ERROR\_DETAILS and retry metadata. Both tables are used for alerting and ServiceNow ticket creation.

# 4. Business Rules

- Only claims with status == 'PENDING' are processed by Function Two.  
- PendedReason should contain 'Prime' or primePended flag true to include claim.  
- Duplicate claims are detected by CLAIM\_GUID/CLAIM\_ID and skipped (logged as INFO).

## 5. Developer Implementation Plan (Phase A–E)

Phase A — Scaffolding:

- Create solution with two Azure Function projects (FunctionOne.Copyer, FunctionTwo.Processor).

- Add shared library: Models (Claim, AuditEntry), Services (BlobService, QueueService, MemberApiClient), Repositories (SnowflakeRepository).

Phase B — Function One development:

- Implement QueueTrigger, message parsing, blob download & copy, audit insert, enqueue staging message.

Phase C — Function Two development:

- Implement QueueTrigger, blob parsing, filtering logic, Member API integration, insert to PENDED\_CLAIMS\_OUTBOUND, and ERROR\_LOG handling.

Phase D — Infra & Deployment:

- Terraform for storage accounts, queues, function apps, managed identities, key vault.  
- Azure DevOps YAML for CI/CD with unit tests, code scanning, and deployment stages.

Phase E — QA & Integration:

- Integration tests with Azurite or sandbox storage account. Load testing and observability verification.

### 5.1 *Developer Checklist & Local Setup*

- Install .NET 6 LTS SDK, Azure Functions Core Tools, Visual Studio/VSCode.  
- Configure local.settings.json with placeholders for StorageConnectionString, KeyVault URI, Member API URL, Snowflake placeholders.  
- Run unit tests with xUnit & Moq.

## 6. Monitoring & Operations

- Application Insights: function traces, exceptions, custom metrics (processed\_count, error\_count).

- Azure Monitor Alerts: high error rate, DLQ depth, Snowflake ingestion failures.

- ServiceNow integration: auto-create ticket for critical failures (DLQ items, Snowflake outage).

## 7. References

- Azure Functions docs, Azure Blob & Queue Storage docs, Snowflake docs, Apigee docs, ServiceNow API docs.

## 8. Amendment History

Version 1.0 — 2025-10-05 — Initial detailed SDD for PCSE.

# Appendix A – Snowflake DDL and Sample Insert

Use the following DDLs to create the required tables in the BCBSRI\_CLAIMS.OUTBOUND schema. Update placeholders for your environment.

CREATE TABLE LANDINGDEV.DOP\_MPS\_PRM.LN\_PSCE\_PENDED\_CLAIMS\_OUTBOUND (

CLAIM\_GUID VARCHAR (36) NOT NULL,

RAW\_PAYLOAD VARIANT,

CLAIM\_PATH VARCHAR (500),

STATUS VARCHAR (50),

ERROR\_CODE VARCHAR (10),

ERROR\_MESSAGE VARCHAR (512),

PROCESSED\_BY VARCHAR (128),

CREATED\_BY VARCHAR (128),

CREATED\_TIMESTAMP TIMESTAMP\_LTZ (9),

CONSTRAINT PK\_PENDED\_CLAIMS\_OUTBOUND PRIMARY KEY (CLAIM\_GUID)

);

*Sample Insert Query with Blob Data*

INSERT INTO

LANDINGDEV.DOP\_MPS\_PRM.LN\_PSCE\_PENDED\_CLAIMS\_OUTBOUND (

CLAIM\_GUID,

RAW\_PAYLOAD,

CLAIM\_PATH,

STATUS,

ERROR\_CODE,

ERROR\_MESSAGE,

PROCESSED\_BY,

CREATED\_BY,

CREATED\_TIMESTAMP

)

VALUES (

'550e8400-e29b-41d4-a716-446655440000',

PARSE\_JSON('{

"claimId": "CLM123456",

"memberId": "MBR98765",

"serviceDate": "2025-09-30",

"amount": 450.00,

"diagnosisCodes": ["D123", "D456"],

"provider": {

"id": "PRV001",

"name": "XYZ Medical Center"

}

}'),

'https://<storage-account>.blob.core.windows.net/<container>/CLM123456.json',

'PENDING',

NULL,

NULL,

'AzureFunctionProcessor',

'BlobIngestionService',

CURRENT\_TIMESTAMP()

);

# References

| **Document Name** | **URL** |
| --- | --- |
| Identify the fields not in the Claims Details API that were in Get Claimdetail service and View Claimdetail service. | [USER STORY 49966:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/49966)  [Missing Mapping V1.xlsx](https://bcbsri.sharepoint.com/sites/APIManagementPlatform-365/Shared%20Documents/Claims%20API%202.1/Missing%20Mapping%20V1.xlsx?d=w3b2ccd956cae4744be82cbfbac4c8580&csf=1&web=1&e=sgltax) |
| Identify gaps in the mapping document from Claims API with EDR & SSDS | [USER STORY 52443:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/52443)  [Missing Mapping V1-03242025.xlsx](https://bcbsri.sharepoint.com/sites/APIManagementPlatform-365/Shared%20Documents/Claims%20API%202.1/Missing%20Mapping%20V1-03242025.xlsx?d=w9a3021a2223f49229fcc59373937b52b&csf=1&web=1&e=RUymbT) |
| Identify fields that are needed by consumers - SFDC and Legacy portal | [USER STORY: 52442:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/52442)  [Missing Mapping V1-05132025.xlsx](https://bcbsri.sharepoint.com/sites/APIManagementPlatform-365/Shared%20Documents/Claims%20API%202.1/Missing%20Mapping%20V1-05132025.xlsx?d=wa684e0376a514e0ea7a584a14c70b937&csf=1&web=1&e=lW23zq) |
| Create modern designs for future Claims API with two options | [USER STORY: 57130:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/57130) |
| Claims API Modernization: Document findings and recommendation for next steps | [USER STORY: 54743:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/54743)  [Claims\_API\_Modernization\_Summary\_V0.1.docx](https://bcbsri.sharepoint.com/sites/APIManagementPlatform-365/Shared%20Documents/Claims%20API%202.1/Claims_API_Modernization_Summary_V0.1.docx?d=wb8e776579cee4c70818b7d649df41647&csf=1&web=1&e=J2iaBa) |
| Claims Details API Redesign | [USER STORY: 15847:](https://dev.azure.com/BCBSRI/cloud-api-modernization/_workitems/edit/15847)  2 SOA Services  [GetClaimDetail](https://bcbsri.sharepoint.com/sites/it_sysops_pub/appsupp/_layouts/15/Doc.aspx?sourcedoc=%7b40355936-F71C-4393-9F56-979C25A4C185%7d&file=GetClaimDetails%20.docx&action=default&mobileredirect=true&DefaultItemOpen=1)  [ViewClaimDetails](https://bcbsri.sharepoint.com/sites/it_sysops_pub/appsupp/_layouts/15/Doc.aspx?sourcedoc=%7b3166B9A8-B84D-4596-82D5-520045E1BAF5%7d&file=View%20Claim%20Details.docx&action=default&mobileredirect=true&DefaultItemOpen=1) |

# Amendment History

| **Version** | **Date** | **Additions / Modifications** | **Prepared / Revised By** |
| --- | --- | --- | --- |
| 1.0 | 10/01/2025 | Initial Version | Saikiran Reddy P |